# PROGRAMACIÓN PARA SISTEMAS

TEMA 3

## Unix: Shell y Scripts



## Unix: Shell y Scripts

### INTRODUCCIÓN

- LOS PUNTOS QUE SE TRATARÁN SON LOS SIGUIENTES:
  - EXPRESIONES CONDICIONALES
  - CONSTRUCCIONES CONDICIONALES
  - CONSTRUCCIONES ARITMÉTICAS
  - BUCLES

**BIBLIOGRAFÍA RECOMENDADA:**

- *The UNIX Programming Environment, B. Kernighan, R. Pike*
- *BASH Cookbook, C. Albing, JP. Vossen, C. Newham*
- *http://www.gnu.org/software/bash/manual/bashref.html*

## Unix: Shell y Scripts

### EXPRESIONES CONDICIONALES

- DAN COMO RESULTADO *verdadero O falso* Y SE EMPLEAN CON LOS COMANDOS DE TEST: (*[[*), (*[* ) , *test*
- PRECEDIDAS DEL OPERADOR (!) INVIERTEN EL RESULTADO:

  *¡ expresion* ES FALSA SI *expresion* ES VERDADERA Y VICEVERSA.
- PUEDEN SER *unarias* o *binarias*, DEPENDIENDO DEL NÚMERO DE ARGUMENTOS QUE SE EMPLEEN

## Unix: Shell y Scripts

### EXPRESIONES CONDICIONALES

- EXPRESIONES UNARIAS USUALES CON FICHEROS:

  *-a file (-e file):* VERDADERO SI EL FICHERO *file* EXISTE
  *-f file* : VERDADERO SI EL FICHERO *file* EXISTE Y ES REGULAR (NO UN *enlace*)
  *-h file* : VERDADERO SI *file* ES UN ENLACE SIMBÓLICO
  *-d file* : VERDADERO SI *file* ES UN DIRECTORIO
  *-r file* : VERDADERO SI *file* TIENE PERMISO DE LECTURA
  *-s file* : VERDADERO SI *file* NO ESTÁ VACÍO
  *-w file* : VERDADERO SI *file* TIENE PERMISO DE ESCRITURA
  *-o file* : VERDADERO SI EL PROPIETARIO ES EL USUARIO
  *-g file* : VERDADERO SI SU GRUPO ES EL DEL USUARIO

## Unix: Shell y Scripts

### EXPRESIONES CONDICIONALES

- EXPRESIONES BINARIAS USUALES CON FICHEROS:

  *file1 -nt file2* : VERDADERO SI *file1* ES MÁS RECIENTE QUE *file2* O *file1* EXISTE Y *file2* NO EXISTE
  *file1 -ot file2* : VERDADERO SI *file1* ES MÁS ANTIGUO QUE *file2* O *file1* NO EXISTE Y *file2* SÍ

## Unix: Shell y Scripts

### EXPRESIONES CONDICIONALES

- EXPRESIONES UNARIAS CON TIRAS DE CARACTERES:

  *-z string* : VERDADERO SI LA LONGITUD DE *string* ES CERO
  *-n string, string* : VERDADERO SI LA LONGITUD NO ES CERO
- EXPRESIONES BINARIAS CON TIRAS DE CARACTERES:

  *string1 = string2*
  *string1 == string2* : VERDADERO SI *string1* Y *string2* SON IGUALES
  *string1 != string2* : VERDADERO SI *string1* Y *string2* SON DISTINTOS
  *string1 < string2* : VERDADERO SI *string1* ES MENOR EN ORDEN ALFABÉTICO QUE *string2*
  *string1 > string2* : VERDADERO SI *string1* ES MAYOR EN ORDEN ALFABÉTICO QUE *string2*

## Unix: Shell y Scripts

### EXPRESIONES CONDICIONALES

- EXPRESIONES BINARIAS CON ENTEROS:

  val1 -eq val2 : VERDADERO SI val1 = val2
  *val1 -ne val2* : VERDADERO SI *val1* ≠ *val2*
  *val1 -lt val2* : VERDADERO SI *val1* < *val2*
  *val1 -le val2* : VERDADERO SI *val1* ≤ *val2*
  *val1 -gt val2* : VERDADERO SI *val1* > *val2*
  *val1 -ge val2* : VERDADERO SI *val1* ≥ *val2*
- EN EXPRESIONES CONDICIONALES ENCERRADAS ENTRE [[ ]], EL RESULTADO ES EL MISMO SI SE EMPLEAN LAS VARIABLES O SU VALOR:

  [[ var1 -eq var2 ]] : VERDADERO SI $var1 = $var2

## Unix: Shell y Scripts

### EXPRESIONES CONDICIONALES

- ALGUNOS EJEMPLOS:

  *var1=""; [ -z $var1 ]; echo $?;* → *[xxx@triqui3 ~] $ 0 (verdadero)*
  *var1=""; [[ -z $var1 ]]; echo $?;* → *[xxx@triqui3 ~] $ 0 (verdadero)*
  *var1=""; test –z $var1; echo $?;* → *[xxx@triqui3 ~] $ 0 (verdadero)*
  *var1="A"; var2="B";*
  *[ $var1 == $var2 ]; echo $?;* → *[xxx@triqui3 ~] $ 1 (falso)*
  *[[ $var2 == $var2 ]]; echo $?;* → *[xxx@triqui3 ~] $ 1 (falso)*
  *test $var2 == $var2; echo $?;* → *[xxx@triqui3 ~] $ 1 (falso)*
  *var1=1; var2=2;*
  *[ $var1 -lt $val2 ]; echo $?;* → *[xxx@triqui3 ~] $ 0 (verdadero)*
  *[[ $var1 -lt $val2 ]]; echo $?;* → *[xxx@triqui3 ~] $ 0 (verdadero)*
  *test $var1 -lt $val2 ; echo $?;* → *[xxx@triqui3 ~] $ 0 (verdadero)*

## Unix: Shell y Scripts

### EXPRESIONES CONDICIONALES

- CON EL COMANDO DE TEST [[ SE PUEDEN EMPLEAR PATRONES EN EXPRESIONES CONDICIONALES DE LA FORMA:

  *string1 == patron1* *string1 != patron1*

  DONDE LOS PATRONES MÁS USUALES SON:

  *** : CUALQUIER TIRA DE CARACTERES, INCLUYENDO EL NULO
  *?* : UN CARÁCTER CUALQUIERA
  *[c1 c2 ...]* : UN CARÁCTER DE LA LISTA
  *[:clase:]* : DENTRO DE UNA LISTA, CUALQUIER CARÁCTER QUE PERTENEZCA A UNA DE LAS CLASES:
  *alnum alpha ascii blank cntrl digit graph lower*
  *print punct space upper word xdigit*

## Unix: Shell y Scripts

### EXPRESIONES CONDICIONALES

- SI ESTÁ HABILITADA LA OPCIÓN *extglob* DEL SHELL MEDIANTE EL COMANDO *shopt –s extglob* SE PUEDEN USAR IGUALMENTE LISTAS DE PATRONES SEPARADOS POR EL CARÁCTER (|):

  ?(lista) : CERO O UNA OCURRENCIA DE UNO DE LOS PATRONES DADOS
  *(lista) : CERO O VARIAS OCURRENCIAS DE LOS PATRONES DADOS
  *+(lista)* : UNA MÁS OCURRENCIAS DE LOS PATRONES DADOS
  *@(lista)* : EXACTAMENTE UNA OCURRENCIA DE LOS PATRONES DADOS
  *!(lista)* : NINGUNO DE LOS PATRONES DADOS

## Unix: Shell y Scripts

### EXPRESIONES CONDICIONALES

- LAS EXPRESIONES CONDICIONALES SE PUEDEN LIGAR MEDIANTE OPERADORES AND Y OR
- CON LOS COMANDOS (*test*) y (*[ ]* ) LA SINTAXIS ES:

  *[ expr1 –a expr2 ]*
  *test expr1 -a expr2* : VERDADERO SI *expr1* Y *expr2* VERDADERAS
  *[ expr1 –o expr2 ]*
  *test expr1 –o expr2* : VERDADERO SI *expr1 O expr2* VERDADERA
- CON EL COMANDO ([[), LA SINTAXIS ES:

  *[[ expr1 && expr2 ]]* : VERDADERO SI *expr1* Y *expr2* VERDADERAS
  *[[ expr1 || expr2 ]]* : VERDADERO SI *expr1* O *expr2* VERDADERA

## Unix: Shell y Scripts

### CONSTRUCCIONES CONDICIONALES

- IF

  LA SINTAXIS DE LA CONSTRUCCIÓN *if* ES:

  *if comando-test; then*
  *comandos consecuentes;*
  [*elif comando-test; then*
  *comandos consecuentes;*]
  [*else comandos alternativos;*]
  *fi*

## Unix: Shell y Scripts

### CONSTRUCCIONES CONDICIONALES

- EJEMPLO

```
A=1; B=1;
if [ $A -gt $B ]; then echo "A>B"
elif [ $B -gt $A ]; then echo "A<B"
else echo "A=B"
fi
```

- COMPROBAR LO ANTERIOR CON LAS CONSTRUCCIONES CONDICIONALES [[ $A -gt $B ]] Y *[[ $B -gt $A ]]*
- ¿QUÉ SUCEDE SI SE EMPLEAN [[ *B -gt A ]]* Y *[[ B -gt A ]]*

## Unix: Shell y Scripts

### CONSTRUCCIONES CONDICIONALES

- CASE

  LA SINTAXIS DE LA CONSTRUCCIÓN *case* ES:

```
case $variable in
patron1 | patron2 | ….) lista de comandos;;
patron3 | ….) lista de comandos;;
patron4) lista de comandos;;
esac
```

- SE EJECUTA LA PRIMERA LISTA DE COMANDOS QUE COINCIDE CON EL PATRÓN

## Unix: Shell y Scripts

### CONSTRUCCIONES CONDICIONALES

- EJEMPLO

```
ANIMAL=caballo
case $ANIMAL in
caballo | perro | gato) echo –n "Cuatro";;
hombre | canguro) echo –n "Dos";;
*) echo –n "Número desconocido de";;
esac
echo " patas"
```

- CON LA OPCIÓN *nocasematch* DEL SHELL HABILITADA (*shopt –s nocasematch*) NO SE DISTINGUEN LAS MAYÚSCULAS DE LAS MINÚSCULAS

## Unix: Shell y Scripts

### CONSTRUCCIONES CONDICIONALES

- EJEMPLO

```
ANIMAL=caballo
case $ANIMAL in
caballo | perro | gato) echo –n "Cuatro";;
hombre | canguro) echo –n "Dos";;
*) echo –n "Número desconocido de";;
esac
echo " patas"
```

- CON LA OPCIÓN *nocasematch* DEL SHELL HABILITADA (*shopt –s nocasematch*) NO SE DISTINGUEN LAS MAYÚSCULAS DE LAS MINÚSCULAS

## Unix: Shell y Scripts

### CONSTRUCCIONES ARITMÉTICAS

- SE EMPLEAN CON EL COMANDO *let* Y (())
- UTILIZAN LOS OPERADORES ARITMÉTICOS:

  id++, id-- : post-incremento, post-decremento
  ++id, –id : pre-incremento, pre-decremento
  +, - : suma, resta
  * , /, % : multiplicación, división, resto
  ** : exponenciación
  =, +=, -=, *= : asignación

- ADEMÁS, HAY EXPRESIONES CONDICIONALES <=, >=, <, > , ==, != QUE SE PUEDEN EMPLEAR ENCERRADAS ENTRE (( )) COMO LAS VISTAS ANTERIORMENTE CON [[ Y [

## Unix: Shell y Scripts

### CONSTRUCCIONES ARITMÉTICAS

- EJEMPLOS:

*let num=0; echo "num=$num"* → *[xxx@triqui3 ~] $ num=0*
*let num=num+1; echo "num=$num"* → *[xxx@triqui3 ~] $ num=1*
*let num+=1; echo "num=$num"* → *[xxx@triqui3 ~] $ num=2*
*(( num=num**2 )); echo "num=$num"* → *[xxx@triqui3 ~] $ num=4*
*(( num>3 )); echo $?* → *[xxx@triqui3 ~] $ 0 (verdadero)*

## Unix: Shell y Scripts

**BUCLES**

- COMANDO *until*:

  *until comando-test; do comandos-consecuentes; done*
  *until [[ num -eq 0 ]]; do { (( num-=1 )); echo $num; }; done*
  *until [ $num -eq 0 ]; do { (( num-=1 )); echo $num; }; done*

- COMANDO *while*:

  *while comando-test; do comandos-consecuentes; done*
  *while [[ num –gt 0 ]]; do { (( num-=1 )); echo $num; }; done*
  *while [ $num –gt 0 ]; do { (( num-=1 )); echo $num; }; done*

## Unix: Shell y Scripts

**BUCLES**

- COMANDO *for*:

  *for variable in lista; do comandos-consecuentes; done*

  *for A in a b c; do echo $A; done*
  *for A in *; do echo $A; done*

- COMANDO *for* (FORMATO ALTERNATIVO):

  *for (( expr1 ; expr2: expr3 )); do comandos-consecuentes; done*
  *for (( i=0; i<10; i++ )); do echo $i; done*